CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

A Rolling Horizon Algorithm for Auto-Carrier Transportation

Jean-François Cordeau
Mauro Dell'Amico
Simone Falavigna
Manuel Iori

June 2014

CIRRELT-2014-27

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

A Rolling Horizon Algorithm for Auto-Carrier Transportation

Jean-François Cordeau[1,*], Mauro Dell'Amico[2], Simone Falavigna[2,3], Manuel Iori[2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Logistics and Operations Management, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[2] DISMI, University of Modena and Reggio Emilia, Via Amendola 2, Padiglione Morselli, 42122, Reggio Emilia, Italy

[3] EMAK S.p.A., Via E. Fermi 4, 42011 Bagnolo in Piano, Reggio Emilia, Italy

Abstract. This paper introduces a rolling horizon algorithm to plan the delivery of vehicles to automotive dealers by a heterogeneous fleet of auto-carriers. The problem consists in scheduling the deliveries over a multiple-day planning horizon during which requests for transportation arrive dynamically. In addition, the routing of the auto-carriers must take into account constraints related to the loading of the vehicles on the carriers. The objective is to minimize the sum of traveled distances, fixed costs for auto-carrier operation, service costs, and penalties for late deliveries. The problem is solved by a heuristic that first selects the vehicles to be delivered in the next few days and then optimizes the deliveries by an iterated local search procedure. A branch-and-bound search is used to check the feasibility of the loading. To handle the dynamic nature of the problem, the complete algorithm is applied repeatedly in a rolling horizon framework. Computational results on data from a major European logistics service provider show that the heuristic is fast and yields significant improvements compared to the sequential solution of independent daily problems.

Keywords. Vehicle routing, loading, auto-carrier, rolling horizon.

Acknowledgements. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant 227837-09 and by CAPES/Brazil under Grant PVE no. A007/2013. This support is gratefully acknowledged.

---

* Corresponding author: Jean-Francois.Cordeau@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2014

1 Introduction

In 2012 the global demand for cars, light commercial vehicles, and trucks amounted to approximatively 82 million vehicles, with an increase of about 5% with respect to 2011 (see, e.g., ANFIA [4]). The majority of the new vehicles sold every year are first delivered by the manufacturers to third-party logistics providers (3PLs). The vehicles are then delivered by these 3PLs to the dealers who sold them, where they can be collected by the final customers. The delivery to the 3PLs is often performed via rail, as it involves a large number of vehicles, a single origin, and a single destination. Delivery to the dealers is instead performed via auto-carriers, as it typically involves a small number of vehicles and several destinations.

Auto-carriers are special trucks, usually composed by a tractor and perhaps a trailer, both equipped with loading platforms. These platforms are used to load the vehicles at the 3PL and unload them at the dealers. The vehicles are not simply loaded straight, but they can be lifted and rotated in several ways by means of special loading equipments. This is done to increase the number of vehicles that can be transported at the same time, thus improving the efficiency of the distribution process. An example of a modern auto-carrier equipped with four loading platforms, two in the tractor and two in the trailer, and carrying eleven vehicles is depicted in Figure 1.

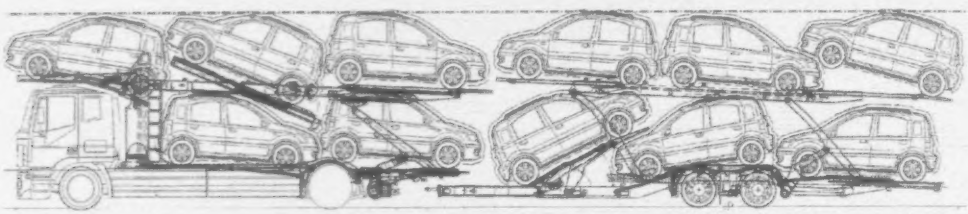

Figure 1: An example of an auto-carrier carrying eleven vehicles (source: Rolfo Spa, Italy).

Once a vehicle is sold by a dealer, the 3PL has a certain range of days, i.e., a *time window*, in which it is supposed to deliver the vehicle to the dealer. If the company fails to perform the delivery within the given time window, it incurs a penalty cost, which is determined by the contract signed between the company and the manufacturer. Some vehicles are sold customized to the customer preferences. In this case the dealer sends an order to the manufacturer, which in turn produces the vehicle and then sends it to the 3PL. As soon as the 3PL receives the vehicle, it contacts the dealer to organize the delivery. Other vehicles are instead sold turnkey, and in this case they are usually already available at the 3PL when the dealer places the order. In any case, the company has to organize the distribution plan for a few days ahead, taking care of demands that arrive dynamically on a daily basis and have to be served soon, possibly by the next day.

Companies operating auto-carriers are obviously interested in optimizing their distribution as this can clearly result in large cost savings. The problem they face is, however, very complex because it involves several constraints and objectives. Typical constraints derive from the fact that an auto-carrier has a limited traveling distance per day and, of course, the vehicles it carries should be feasibly loaded on its platforms. A *Last-In First-Out* (LIFO) policy (also known as *sequential loading* policy, see, e.g., Cordeau et al. [9]) is usually imposed on the loading, i.e., when visiting a dealer, the vehicles it requires should be unloaded without moving vehicles destined to other dealers.

We note that the fleet that can be used for the deliveries is typically heterogenous, as it involves auto-carriers having different costs and loading capacities. Furthermore, due to the fact that long-distance routes may last more than one day, the size and composition of the available fleet varies day by day, depending on the deliveries carried out on previous days. We also note that typical

objective functions include fixed costs for using the auto-carriers, traveling costs for performing the deliveries, service costs for visiting the dealers, and penalties for violations of the time windows.

Because of their importance in many markets, auto-carrier distribution problems have received a good level of attention in the past. Agbegha [1] and Agbegha, Ballou and Mathur [2] focused on the subproblem of loading vehicles in the auto-carrier. They divided the auto-carrier in a fixed number of slots, and modeled it by using a loading network where each vertex corresponds to a slot. The same model was computationally evaluated on a set of random instances by Lin [18].

Tadei, Perboli and Della Croce [24] studied the maximization of profits over a multiple-day horizon for an Italian distribution company. They proposed a heuristic based on an integer linear programming formulation, in which the routing problem was relaxed by grouping all possible destinations into clusters. They also relaxed the loading problem as follows: (i) they computed for each auto-carrier an *equivalent auto-carrier length*, taking into account the auto-carrier loading equipments; (ii) they computed for each vehicle an *equivalent vehicle length*, taking into account the vehicle shape; (iii) they modeled the loading as a single capacity constraint, imposing the sum of the equivalent vehicle lengths to be no larger than the equivalent auto-carrier length. A similar heuristic was used by Cuadrado and Griffin [10] to solve a distribution case in Venezuela.

Miller [20] proposed a greedy heuristic and some simple local search procedures for a case arising in the USA market. He modeled the auto-carrier as two flat loading platforms and loaded the vehicles straight on the platforms. He studied the single-day distribution and used simple heuristics to design the routes. The USA market was also studied by Jin et al. [17], who developed a business scheme to evaluate and compare the transportation costs via roads and via railway.

The most recent approach, as far as we know, was developed by Dell'Amico, Falavigna and Iori [11], who proposed a heuristic algorithm for a real-world distribution case arising in the Italian market. They developed a branch-and-bound algorithm for the loading subproblem and an iterated local search (ILS) heuristic for the overall problem. Their ILS starts with a greedy heuristic solution which is iteratively perturbed and improved by the use of local search operators. The branch-and-bound is invoked to check the feasibility of the loadings considered during the ILS process.

The algorithm of Dell'Amico, Falavigna and Iori [11] is the first one that simultaneously provides detailed solutions of both the loading and routing components of the auto-carrier transportation problem, but it can only solve a static, single-day problem. In this paper we build upon this algorithm, but extend it to the more realistic case of multiple-day distribution and dynamic demands, also including new operational constraints and additional cost components in the objective function. More precisely, we introduce a heuristic algorithm that, given partial information on the demands, plans not only the routes and loadings for the first day, but also those for the next few days. The distribution plan for the first day is then implemented: vehicles scheduled for delivery are loaded on the auto-carriers, which are then sent to perform their routes. On the next day, the availability of the fleet of auto-carriers is updated, the new dynamically revealed demands are added to those that have not been served yet, and the process is iterated in a so-called *rolling horizon* fashion.

The problem is motivated by the activity of Gruppo Mercurio, a large 3PL that delivers cars, commercial vehicles, and trucks for almost all of the major manufacturers. It operates in the Italian market, where it is the leading company, and in the European market, with a large fleet of auto-carriers.

The main contributions of this paper are the following:

- we introduce a new optimization problem that models an important real-world distribution process;
- we adapt the ILS from [11], developed for the static single-day problem, to the new, more

realistic, dynamic, multiple-day problem, obtaining a heuristic that is more suitable for the solution of real-world auto-carrier transportation;

- we develop new local search procedures, and propose new algorithms to balance demands and vehicle usage, so as to obtain the best possible solutions in a dynamic setting;
- we present extensive computational tests on a set of instances derived from the real-world distribution case faced by Gruppo Mercurio in Italy;
- we perform a large computational sensitivity analysis, obtaining interesting insights in the performance of the algorithm and in the difficulty of the problem.

The remainder of the paper is organized as follows. In Section 2 we give a formal description of the problem that we are addressing. In Section 3 we describe the ILS heuristic, paying particular attention to the dynamic part and to the mechanisms required to run the algorithm in a rolling horizon environment. In Section 4 we present computational results, and finally in Section 5 we draw some conclusions. For the sake of clarity, we note that in the next sections we will use *vehicle* to denote an item to be transported (e.g., a car, a truck, or a van), *auto-carrier* to denote a truck used to transport the vehicles, and *dealer* to denote a customer requiring the delivery of one or more vehicles.

2 Problem Description

We are given a complete graph $G = (N, E)$, where the set of vertices $N = \{0, 1, \ldots, n\}$ is partitioned into vertex 0, corresponding to the depot, and vertices $\{1, 2, \ldots, n\}$, corresponding to the n dealers to be served. Every edge $(i, j) \in E$ is associated with a traveling cost c_{ij}, measured in kilometers (km). We study a distribution process that involves a sequence of D days denoted by $1, 2, \ldots, D$. As usual in dynamic contexts, not all the demands are known in advance, but they are instead revealed day after day. New demands received until the beginning of day d may be served either on day d or on the following days. We denote by M the total set of demands (i.e., vehicles to be delivered) known at the time when the problem is solved. Each dealer $i \in N \setminus \{0\}$ demands a set M_i of vehicles. Each vehicle $k \in M_i$ is associated with a weight w_k, a certain vehicle model (i.e., it has a known shape, used to determine a feasible loading in the way described below), and a time window $[s_k, e_k]$ specifying the interval of days during which the delivery of k should be performed.

The company cannot deliver vehicle k before s_k, and pays a penalty if it delivers the vehicle after e_k. The penalty cost increases when the delay gets larger, according to a policy that depends on a commercial agreement. We use $T_k = \max\{0, d - e_k\}$ to denote the tardiness of the delivery, i.e., the delay of a route that reaches the dealer i on day d, and $\pi(T_k)$, for $T_k > 0$, to denote the resulting penalty cost for the vehicle incurring the delay.

Note that in the case where several vehicles are demanded by the same dealer, it could be impossible or inconvenient to deliver all vehicles by means of a single truck. Consequently, the problem that we study allows to *split* a dealer demand into more than one route. However, a service cost σ is associated with each visit to a dealer, and this tends to lower the overall number of visits.

To perform the deliveries, a heterogeneous fleet of auto-carriers is available. The fleet is composed by a set T of auto-carrier types, where each type $t \in T$ has a maximum weight capacity W_t, a given loading space (see below for details), and a fixed cost C_t. All auto-carriers may perform at most one route per day and may travel for at most L km per day. At the beginning of day 1, K_t auto-carriers are available at the depot for each type t, but this number may vary in the following days, due to the fact that the routes may last more than one day.

We use $\langle R, S, t, d \rangle$ to define a route, where: $R \subseteq N \setminus \{0\}$ denotes the sequence of dealers to be visited; $S_i \subseteq M_i$ gives the set of vehicles to be delivered to each dealer $i \in R$, and $S = \cup_{i \in R} S_i$ is the complete set of vehicles loaded in the auto-carrier; t is the type of auto-carrier used; and d is the day on which the route starts. A route $\langle R, S, t, d \rangle$ thus starts from the depot on day d, visits the dealers in the order specified by R, performing the deliveries of the vehicles in S, and returns empty to the depot on day d or later. The length of a route is simply given by the sum of the c_{ij} values associated with the edges traveled by the route. A route having length $\overline{\ell}$ lasts for $\overline{d} = \lceil \overline{\ell}/L \rceil$ days. An auto-carrier leaving the depot on day d to perform a route that lasts $\overline{d}$ days will be available again to perform another route on day $d + \overline{d}$.

A route $\langle R, S, t, d \rangle$ is said to be *load-feasible* if $\sum_{k \in S} w_k \leq W_t$ and there exists a feasible loading of the vehicles in S on auto-carrier t. The latter constraint means that vehicles must be completely supported by the loading platforms, should be completely contained within maximum cargo length and height (specified by the regulation), and the LIFO unloading policy is respected: when visiting dealer i, all vehicles in S_i can be unloaded directly from the auto-carrier, without moving vehicles destined to dealers to be visited later. To check the loading constraint we make use of the method developed in [11]. This algorithm considers the shapes of the vehicle models and the equipments of the auto-carrier to compute, respectively, the *equivalent vehicle lengths* and the *equivalent platform capacities*. This approximation is accurate in practice and allows to solve all the real life instances that we tested with the operators of the 3PL. Then, it constructs an enumeration tree that attempts to assign the vehicles one at a time to the platfo ns of the auto-carrier without violating capacities, until a feasible loading is found, if any. Lower bounds are used to speed up the process. The resulting algorithm is a branch-and-bound search that is very fast and provides solutions that are accurate in practice.

The objective function that we aim to minimize is given by

$$z_{tot} = z_{km} + z_{fix} + z_{ser} + z_{pen}, \tag{1}$$

where: z_{km} is the sum of the lengths of the performed routes, computed as the sum of the c_{ij} values on the selected edges; z_{fix} is the sum of the fixed costs C_t associated with the use of the auto-carriers; z_{ser} is the total service cost computed by multiplying σ by the total number of visits to dealers; and z_{pen} is the penalty cost computed by summing all the $\pi(T_k)$ penalties for the late deliveries.

The *Dynamic Multi-Period Auto-Carrier Transportation Problem* (DMPACTP) calls for the determination of a set of load-feasible routes to be performed on each of the D days of the planning horizon, with the aim of minimizing the total cost given by (1).

The DMPACTP belongs to the family of combined routing and loading problems, for which we refer the interested reader to the surveys of Iori and Martello [14, 15]. The problem is also related to the periodic VRP in which a multiple-day planning horizon is considered and each customer must receive one or several visits on different days (see, e.g., Francis and Smilowitz [12]). With respect to the problem studied in [11], we introduce the complicating aspects of dynamic demands and multi-period rolling horizon optimization.

In a rolling horizon framework, the problem is solved repeatedly on a reduced planning horizon that shifts forward each time the algorithm is applied. This approach is popular in dynamic vehicle routing problems, where new information becomes available while the vehicle routes are being executed (see, e.g., Mitrović-Minić, Krishnamurti, and Laporte [21], Pillac et al. [22], and Psaraftis [23]). Our problem differs from most other dynamic vehicle routing problems in the sense that vehicle routes cannot be changed as new requests are received. The problem is nevertheless dynamic because some information, i.e., future demands, is unknown when making the initial transportation plan.

Our problem is in fact closely related to the Dynamic Multiperiod Vehicle Routing Problem (DMVRP), first introduced by Angelelli, Speranza, and Savelsbergh [6], in which service requests received at the beginning of a time period must be served either in that period or in the next one. The authors have performed a competitive analysis for simple decision rules in the case of a single vehicle. A generalization of this problem to the case of multiple vehicles was later studied by Angelelli et al. [5] who also considered on-line requests that can arrive while the vehicles are traveling. Wen et al. [25] have studied another variant where the number of periods during which a request can be served can vary between customers. They have introduced a three-phase rolling horizon heuristic in which the first phase consists in selecting the customers to be visited in the next τ days, where τ is the length of the rolling horizon. The second phase then constructs vehicle routes for each day by solving a periodic vehicle routing problem. Finally, the routes to be implemented on the first day are further improved by a tabu search heuristic in the third phase. Very recently, Albareda-Sambola, Fernández, and Laporte [3] have studied a DMVRP in which some probabilistic information regarding future demands is available. The authors have introduced an adaptive service policy to estimate the best time period to serve each request by solving a prize collecting VRP.

Rolling horizon algorithms are also popular in the field of inventory routing, where vehicle routing and inventory decisions must be made concurrently to minimize the cost of distributing products to customers over a multiple-period planning horizon (see, e.g., Bard et al. [7] and Jaillet et al. [16]). Finally, a rolling horizon heuristic was also developed by Bostel et al. [8] for the routing and scheduling of technicians visiting customers for maintenance or service activities.

3 A Rolling Horizon Iterated Local Search

The sketch of the approach that we use to optimize the DMPACTP is given in Algorithm 1. At the beginning of the period, the set M of known demands (vehicles to be delivered) is given, and an initial auto-carrier fleet is available at the depot. On day d, M is updated by including the demands dynamically revealed at the beginning of that day. Then, the algorithm performs two main steps. First, it selects the subset of vehicles to be delivered in the next Δ days $(d, d+1, \ldots, d+\Delta-1)$, denoted by $M'(\Delta)$. Second, it optimizes the routing plans for the corresponding days by invoking a meta-heuristic algorithm, namely an ILS.

The solution returned by the meta-heuristic contains detailed routing plans for each of the Δ days. The routing plan for day d is then implemented, and, consequently, the set of vehicles scheduled for delivery on day d is removed from M, and the available auto-carrier fleet for the next days is updated. The process is repeated for each day in the planning horizon. In practice, a company using this approach would run the steps 3-8 of Algorithm 1 at the beginning of each day. In the following, we give the details of each step in the approach.

Algorithm 1 Rolling Horizon Iterated Local Search

1: Input: set M of known demands and initial auto-carrier fleet
2: **for** (each day d) **do**
3: Include in M the demands revealed at the beginning of day d
4: Select the set $M'(\Delta)$ of vehicles to be delivered in days $d, d+1, \ldots, d+\Delta-1$
5: Optimize deliveries of $M'(\Delta)$ by Iterated Local Search
6: Implement solution for day d
7: Remove from M the vehicles scheduled for delivery on day d
8: Update auto-carrier fleet availability for next days
9: **end for**

Algorithm 2 Vehicle Selection

1: Input: the current set of vehicles M
2: Output: a set $M'(\Delta)$ of vehicles to be delivered in days $d, d+1, \ldots, d+\Delta-1$
3: Initialize $M'(\Delta) = \emptyset$ and $\overline{M} = M$
4: **for** $(k = 1, 2, \ldots, \Delta)$ **do**
5: Move from $\overline{M}$ to $M'(\Delta)$ the m_{urg} most urgent orders, if any
6: Let I_{urg} be the set of dealers requiring the m_{urg} orders
7: Move from $\overline{M}$ to $M'(\Delta)$ at most $\overline{m}$ orders for each dealer in I_{urg}
8: Let I_{prox} be the set of dealers close to those in I_{urg}
9: **repeat**
10: Randomly select a dealer $\bar{i} \in I_{prox}$, if any
11: Move from $\overline{M}$ to $M'(\Delta)$ at most $\overline{m}$ orders required by $\bar{i}$
12: **until** $(k \cdot m_{\max} \leq |M'(\Delta)| \leq k \cdot m_{\max} \cdot (1+\rho)$ **or** $I_{prox} = \emptyset)$
13: **if** $\overline{M} = \emptyset$ **then** break
14: **end for**

The procedure used to select the vehicles to be delivered (step 4 of Algorithm 1) is given in Algorithm 2. This procedure should select a subset of vehicles whose deliveries are geographically correlated, so as to facilitate solutions with low costs, but it should also ensure that the distribution effort is evenly spread over the entire horizon, so as to avoid peaks of deliveries that could result in high costs and unnecessary penalties. To this aim we make use of a series of parameters. In particular, we denote by $m_{\max}$ the number of vehicles that we aim to deliver in a day and by ρ an acceptable tolerance for possibly exceeding $m_{\max}$. We also define a minimum threshold distance $c_{\min}$ and fixed parameters $m_{urg} \leq m_{\max}$ and $\overline{m} \leq m_{\max}$. Finally, we let $\overline{M}$ denote the set $M \setminus M'(\Delta)$.

The procedure first selects the m_{urg} vehicles having the lowest value of e_k, breaking ties randomly, and moves them from $\overline{M}$ to $M'(\Delta)$. It then builds I_{urg} as the set of dealers that requested at least one of the m_{urg} selected vehicles. It considers the dealers in I_{urg} one at a time, in random order, and for each of them it selects up to $\overline{m}$ unscheduled vehicles and moves them from $\overline{M}$ to $M'(\Delta)$. It then builds the set I_{prox} of dealers that are geographically close to those in I_{urg}, by setting $I_{prox} = \{i \in N \setminus I_{urg} : \min_{j \in I_{urg}} c_{ij} \leq c_{\min}\}$. It randomly selects a dealer $\bar{i}$ in I_{prox} requiring one or more of the vehicles in $\overline{M}$, and moves up to $\overline{m}$ vehicles required by $\bar{i}$ from $\overline{M}$ to $M'(\Delta)$. The process is repeated by checking the size of the set being constructed. After processing any step aimed at filling $M'(\Delta)$ with vehicles, the procedure checks if $M'(\Delta)$ reached the desired cardinality. At a given iteration k, it stops as soon as $k \cdot m_{\max} \leq |M'(\Delta)| \leq k \cdot m_{\max} \cdot (1+\rho)$. At the last iteration, we thus try to obtain a set whose cardinality satisfies $\Delta \cdot m_{\max} \leq |M'(\Delta)| \leq \Delta \cdot m_{\max} \cdot (1+\rho)$. Obviously, the procedure also stops if no more vehicles have to be delivered, i.e., if $\overline{M}$ is empty. In this case, the cardinality of $M'(\Delta)$ would be smaller.

On the basis of preliminary computational experiments, we have set $m_{urg} = 10$, $\overline{m} = 50$, $c_{\min} = 15$ km, and $\rho = 0.02$. The value of $m_{\max}$ was set to the average number of vehicles delivered daily by the company that provided us with the data (namely, 774). The reason for using the tolerance ρ is the following: to obtain solutions with a low travel distance, it could appear appropriate to include all vehicles requested by a selected dealer i in $M'(\Delta)$, even in case of a large demand. However, this could result in a too large number of daily deliveries, and thus in an unbalanced distribution plan for the period. For this reason we include vehicles from i in $M'(\Delta)$ even if this results in an excess of $m_{\max}$, but we only accept a slight excess limited by ρ.

The meta-heuristic algorithm that we adopted to optimize the routing plan is a classical ILS (see, e.g., Lourenço, Martin, and Stützle [19]), and its pseudo-code is given in Algorithm 3. It receives

as input the set of deliveries to be scheduled in the first Δ days and the available auto-carrier fleet, and returns a detailed distribution plan.

Algorithm 3 Iterated Local Search

1: Input: the set $M'(\Delta)$ of vehicles and the available auto-carrier fleet
2: Output: a routing plan for days $d, d+1, \ldots, d+\Delta-1$
3: Initialize the incumbent solution by a greedy heuristic and local search procedures
4: **while** (time limit not reached) **do**
5: Perturb the incumbent solution
6: Optimize the perturbed solution using local search procedures
7: Possibly update the incumbent solution
8: **end while**

The search starts by generating an initial heuristic solution by means of a quick greedy heuristic that operates as follows. We choose the first day d in the period, randomly select an auto-carrier t available on day d and a dealer i whose demand set M_i contains one or more vehicles that can be delivered in d. We load t with the unserved vehicles from M_i, one at a time, as long as the load remains feasible. If all vehicles from M_i are loaded, then we look for the dealer nearest to i and re-iterate in the same way until the auto-carrier is fully loaded. We then re-iterate with a new route until all auto-carriers have been used or $m_{\max} \cdot (1+\rho)$ vehicles have been scheduled for delivery on day d. We then proceed to the next day and re-iterate. Note that, because of the limited number of available auto-carriers, the greedy heuristic may return a solution that does not deliver all the vehicles in $M'(\Delta)$. Such a solution is passed anyway to the local search procedures, which try to load the unscheduled vehicles as soon as an auto-carrier is available. In this sense our approach generalizes the one in [11], which only accepts full-delivery solutions.

Several local search procedures are then invoked, one after the other, to improve the solution. Each of them operates in a first-improvement fashion. The use of the first improvement policy leads to a small number of calls to the load feasibility checking procedure, hence making the search less computationally demanding. When no improvement is found, the next procedure is executed, and the process terminates when none of the procedures finds an improvement. In the literature, this is also sometimes referred to as variable neighborhood descent (see, e.g., Hansen, Mladenović, and Moreno Pérez [13]). Every time a local search finds a profitable move, then the feasibility of the loads of the routes involved in the move is checked using the branch-and-bound algorithm of [11]. The following ten local search procedures have been implemented:

- intra-route move: select a dealer and change its order of visit in its current route;
- 1-0 dealer-move: select a dealer i in route $\langle R^a, S^a, t^a, d^a \rangle$ and move its subset S_i^a of vehicles in the lowest-cost position of another route;
- 1-1 dealer-swap: select $i1$ in $\langle R^a, S^a, t^a, d^a \rangle$ and $i2$ in $\langle R^b, S^b, t^b, d^b \rangle$, and swap S_{i1}^a with S_{i2}^b;
- 2-1 dealer-swap: select two dealers $i1$ and $i2$ in $\langle R^a, S^a, t^a, d^a \rangle$ and another dealer $i3$ in $\langle R^b, S^b, t^b, d^b \rangle$, and swap $S_{i1}^a \cup S_{i2}^a$ with S_{i3}^b;
- 1-1 model-swap: consider two dealers $i1$ in $\langle R^a, S^a, t^a, d^a \rangle$ and $i2$ in $\langle R^b, S^b, t^b, d^b \rangle$, select a subset $Q_{i1}^a \subseteq S_{i1}^a$ made by vehicles of the same model, and swap Q_{i1}^a with S_{i2}^b;
- 2-1 model-swap: consider $i1$ and $i2$ in $\langle R^a, S^a, t^a, d^a \rangle$ and $i3$ in $\langle R^b, S^b, t^b, d^b \rangle$, select subsets $Q_{i1}^a \subseteq S_{i1}^a$ and $Q_{i2}^a \subseteq S_{i2}^a$ made by vehicles of the same model, and swap $Q_{i1}^a \cup Q_{i2}^a$ with S_{i3}^b;

- auto-carrier interchange: consider a route $\langle R^a, S^a, t^a, d^a \rangle$ visiting a dealer i whose demand M_i is currently split into two or more routes, and select an auto-carrier t_b having capacity larger than that of t^a, if any. Replace t_a by t_b, and load t_b with S^a and the other subsets of M_i belonging to other routes, one at a time, as long as the load is feasible;

- route addition: select a dealer i whose demand M_i is currently split. Initialize a new route, remove the subsets of M_i from their current routes and load them in the new route, one at a time, as long as the load is feasible;

- route swap: select two routes $\langle R^a, S^a, t^a, d^a \rangle$ and $\langle R^b, S^b, t^b, d^b \rangle$ with $d^a \neq d^b$, and exchange d^a with d^b;

- route insertion: if some orders in $M'(\Delta)$ are not scheduled yet, select an available auto-carrier, if any, and compute the first day in the interval in which *i)* the auto-carrier is available at the depot, and *ii)* a delivery of the unscheduled vehicles could happen. Create a new route by loading the unscheduled vehicles in the auto-carrier using the greedy heuristic already used at step 3 of Algorithm 3.

The first eight procedures are derived from [11], but generalized to take into consideration the time windows for the deliveries, the penalty costs for the late deliveries, and the fixed costs for using the auto-carriers. The last two procedures are new. With the exception of intra-route move and route insertion, all procedures involve the optimization of two or more routes at a time. Some of them, such as 1-1 model-swap and 2-1 model-swap, may increase the number of split deliveries, while others, such as auto-carrier interchange and route addition, attempt to reduce this number. Procedure route swap does not affect the traveling costs, but has the purpose of reducing the total cost of penalties. Procedure route insertion increases the number of auto-carriers used, and hence the resulting auto-carrier usage cost. It is, however, very important to obtain feasible solutions. In our tests the initial greedy sometimes failed to load the entire set of vehicles in the available auto-carriers but, after some iterations, the local search procedures managed to obtain more compact loads and consequently free some auto-carriers for one or more of the Δ days. These auto-carriers are thus used by route insertion to try to obtain a solution that delivers all the vehicles in $M'(\Delta)$.

The perturbation method adopted in our ILS (step 5 of Algorithm 3) is obtained by randomly selecting a dealer i, finding all dealers whose distance from i is not larger than a given threshold γ, and then removing all routes currently serving these dealers. Then, we invoke the greedy heuristic again to assign the removed demands to the available auto-carriers. The perturbed solution is then given to the set of local search algorithms. If a feasible solution of lower cost is found, then the incumbent is updated. The value of γ was set to 200 km on the basis of preliminary tests.

After a certain time limit is reached, the ILS is halted and the incumbent solution that it found is returned to the outer rolling horizon loop. The ILS optimizes costs, but may return solutions in which some routes carry a small load of vehicles. Thus, the routes scheduled for day d are quickly checked, and, if their load is smaller than 80% of their loading capacity, they are rejected. The routes that are not rejected are implemented for the day (step 6 of Algorithm 1). Consequently, the vehicles to be delivered are erased from M and the auto-carrier fleet available in the next days is updated.

The effectiveness of the proposed algorithm is assessed by extensive computational experiments described in the next section.

4 Computational Results

The algorithm was coded in C++ and run on a 2.7 Ghz Pentium Dual-Core processor with 2 GB RAM, under the Windows XP operating system. It was tested on a benchmark set of instances based on the actual vehicle deliveries performed by Gruppo Mercurio during the 23 working days of July 2009. These instances contain 17,804 vehicles to be delivered, corresponding to 723 standard vehicle models from several car manufacturers. They involve a fleet composed by two auto-carrier types, the first having four loading platforms and a weight capacity of 15.1 tons, and the second having two loading platforms and a weight capacity of 6 tons. The depot is located near the city of Parma, in northern Italy.

The instances are publicly available at `www.or.unimore.it/A-CTP`, and were already used to test the ILS in [11]. This ILS operates without a rolling horizon perspective, *sequentially* solving one instance after the other, and is used, in Section 4.1, as a comparison basis to evaluate the new algorithm described in this paper. In the available instances, all vehicles are assigned to a given day (the day on which the delivery was actually performed by the company) and, unfortunately, the information on the original time window in which the delivery could take place has not been stored by the company. For this reason we adapted the instances to fit the DMPACTP definition by generating the missing information in a way that mimics what happens in real-world operations. Let $\overline{d}_k$ be the day on which vehicle k was delivered, and recall that $[s_k, e_k]$ denotes the time window in which the delivery of k should be performed. We set $s_k = \overline{d}_k$ with 40% of probability, $s_k = \overline{d}_k - 1$ with 30%, $s_k = \overline{d}_k - 2$ with 20%, and $s_k = \overline{d}_k - 3$ with 10%. We then set $e_k = s_k + 3$ if the dealer requiring vehicle k is located in the northern part of Italy, $e_k = s_k + 4$ if it is in the central part of Italy, and $e_k = s_k + 5$ if it is in the southern part. Note that in this way $\overline{d}_k \in [s_k, e_k]$, so the original delivery day would incur no penalty in the modified instances. In the first day of the planning horizon, the fleet is composed of 130 auto-carriers of type 1 and seven auto-carriers of type 2. All auto-carriers can travel for at most $L = 560$ km per day.

For the objective function components, the costs c_{ij} are set to the minimum travel distance in km on the real road network. The other costs derive directly from the activity of Gruppo Mercurio, and are converted into km so as to obtain a uniform evaluation. In particular, the fixed auto-carrier costs are set to $C_1 = 200$ for the larger auto-carriers and $C_2 = 150$ for the smaller ones. The service cost σ for each visit to a dealer is set to 30. For delay penalties, recall that $T_k = \max\{0; d - e_k\}$ denotes the delay of the delivery of vehicle k performed on day d. The penalty function $\pi(T_k)$, to be paid for each vehicle k delivered late, follows this scheme: $\pi(T_k) = 9$ if $T_k \leq 2$, $\pi(T_k) = 13$ if $3 \leq T_k \leq 5$, and $\pi(T_k) = 20$ if $T_k \geq 6$.

In Section 4.1, to compare with the existing literature, we run our algorithm with a limited objective function that takes into consideration only travel distance and penalties, whereas in Section 4.2 we consider the complete objective function. In both cases we perform a large series of sensitivity analyses to gain insight in the computational behavior of the proposed algorithm.

4.1 Comparison with the Sequential Solution Approach

In the following we use RH-ILS to denote the rolling horizon algorithm described in Section 3, and Seq-ILS to denote the sequential ILS introduced in [11]. The two algorithms have been run on the same computer. The tests for Seq-ILS involved the minimization of just the traveling costs z_{km} (see the objective function (1)), by assuming that each vehicle is delivered on the same day as in the original schedule. To have a fair comparison, we considered in this section a version of the RH-ILS that disregards z_{fix} (fixed auto-carrier costs) and z_{ser} (service costs). We kept z_{pen} (penalty costs for late deliveries) because otherwise the algorithm would completely disregard the time windows,

obtaining solutions that are very convenient for z_{km} but unrealistic with respect to the original delivery dates. Nevertheless, penalty costs are not included in the figures reported below.

We ran our ILS algorithm several times, by varying the number Δ of days that are considered at every iteration. On the one hand, large Δ values allow a deeper evaluation of the solution space but impose a larger computational burden for the local search procedures. On the other hand, small Δ values require less computing effort, but reduce the number of potential solutions explored. For this reason it is important to perform a sensitivity analysis that combines Δ and the computational time limit (denoted by tl) given to the ILS.

The ILS has been run by considering five values of Δ (namely, 1, 2, 3, 4, and 5) and five values of tl (namely, 1, 3, 5, 10, and 25 minutes of CPU time). A time limit of, say, one minute means that the ILS runs for one minute on each of the 23 days in the planning horizon. The results of this test are depicted in Figure 2. The graph in the top part of the figure gives on the y-axis the average z_{km} value over the 23 instances, and on the x-axis the adopted Δ value. The graph in the bottom part reports the same values, but with a different emphasis: on the y-axis the average z_{km} value, and on the x-axis the adopted time limit. The five versions of our RH-ILS are compared with the average result obtained by the original Seq-ILS.

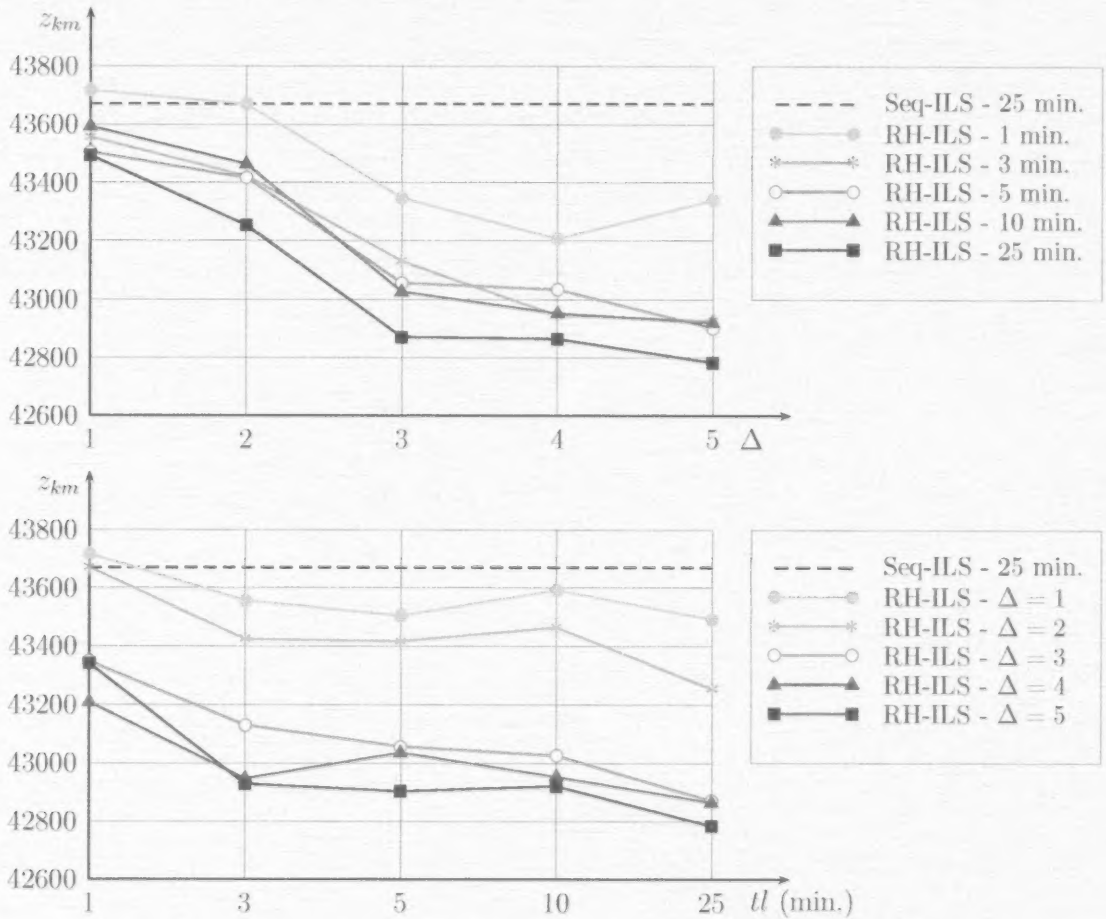

Figure 2: Comparison of the average z_{km} values obtained by the Seq-ILS in [11] and the new RH-ILS, related to Δ (above) or time limit (below).

Figure 2 shows that only the ILS configuration with $tl = 1$ and $\Delta = 1$ is worse on average than

the original Seq-ILS, whereas all other configurations produce improvements in the value of z_{km}. For the ILS with $tl = 1$, the case with $\Delta = 4$ is the one giving the best result, showing that $\Delta = 5$ is too large for such a small time limit. For all other time limits, the use of $\Delta = 5$ is slightly better than $\Delta = 4$ and $\Delta = 3$, which in turn are much better than $\Delta = 2$ and $\Delta = 1$. We can conclude that the rolling horizon algorithm is much more effective than the sequential one, because it allows to obtain better quality solutions with a much smaller computational effort. The best configuration, i.e., the one having $tl = 25$ minutes and $\Delta = 5$, improves the average z_{km} value by about 900 km per day, leading to savings of more than 20,000 km for the full planning horizon.

Figure 3 provides a possible explanation for this good performance, comparing the best RH-ILS configuration against Seq-ILS and showing results for each day. The top part of the figure presents the number n_{ac} of auto-carriers used for each day in the best solutions found by the two algorithms. Seq-ILS uses a large number of auto-carriers at the beginning of the period (117 on Jul-02) and decreases this number consistently towards the end (just 30 on Jul-31). Conversely, the RH-ILS uses between 68 and 103 auto-carriers, obtaining a more balanced distribution and a better use of the available fleet.

The middle part of Figure 3 gives the number n_{del} of vehicles delivered to dealers for each algorithm and each day. The behavior is similar to that noted for the number of auto-carriers, with the Seq-ILS delivering between 272 and 1139 vehicles per day, and the RH-ILS between 650 and 989. The bottom part of the figure shows the number n_{vis} of visits to dealers. The difference between the two algorithms is in this case even more substantial. The Seq-ILS performs on average 240 visits (between 104 and 297 per day), whereas the RH-ILS only 189 (between 151 and 228). It is obvious that the rolling horizon algorithm is capable of reducing the number of split demands, delivering the same number of vehicles with fewer visits to dealers, thus obtaining a lower cost and a better service to the dealers.

4.2 Evaluation of the Rolling Horizon Algorithm

In this section we evaluate our RH-ILS algorithm by using the complete objective function defined in (1), and again testing different values for the time limit tl and for Δ. In Figure 4 we show the evolution of the average objective function values obtained by the attempted RH-ILS configurations, related to the value of Δ (above) and that of tl (below).

In the top part of Figure 4 we can notice that the values obtained with $\Delta = 1$ are pretty high, being always above 66,000. They then decrease consistently with $\Delta = 2$ and even more with $\Delta = 3$. The best average value is obtained with $\Delta = 3$ for $tl = 1$ but with $\Delta = 4$ for $tl = 3$, 5, or 10 minutes. The configuration with $\Delta = 5$ is the best only when the RH-ILS is allowed to run for 25 minutes. By comparing with Figure 2, we can conclude that the use of the complete objective function (z_{tot} instead of z_{km}) increases the complexity of the problem and imposes a higher computational effort to the local search algorithms. We can also conclude that $\Delta = 4$ is a good value for this set of instances when only moderate CPU times are allowed.

Similar observations apply to the bottom part of Figure 4. The configuration with $\Delta = 1$ has quite poor results, and the one with $\Delta = 2$ is slightly better. The ones with $\Delta = 3$, 4, or 5 are again better and quite similar among them. Note that when $\Delta = 1$ or $\Delta = 2$ the RH-ILS can obtain better average z_{tot} values with smaller tl values than with larger ones. This is due to the fact that finding a low cost solution for a certain day could increase the solution costs for the next days, thus yielding a higher overall cost. The algorithm behavior appears thus quite myopic for small values of Δ. The myopic behavior then disappears for larger values of Δ, which are thus preferable. The case with $\Delta = 4$ is better than $\Delta = 5$ for all attempted time limits except 25 minutes. Overall, a time limit of 10 minutes seems to be a good compromise between solution quality and computational

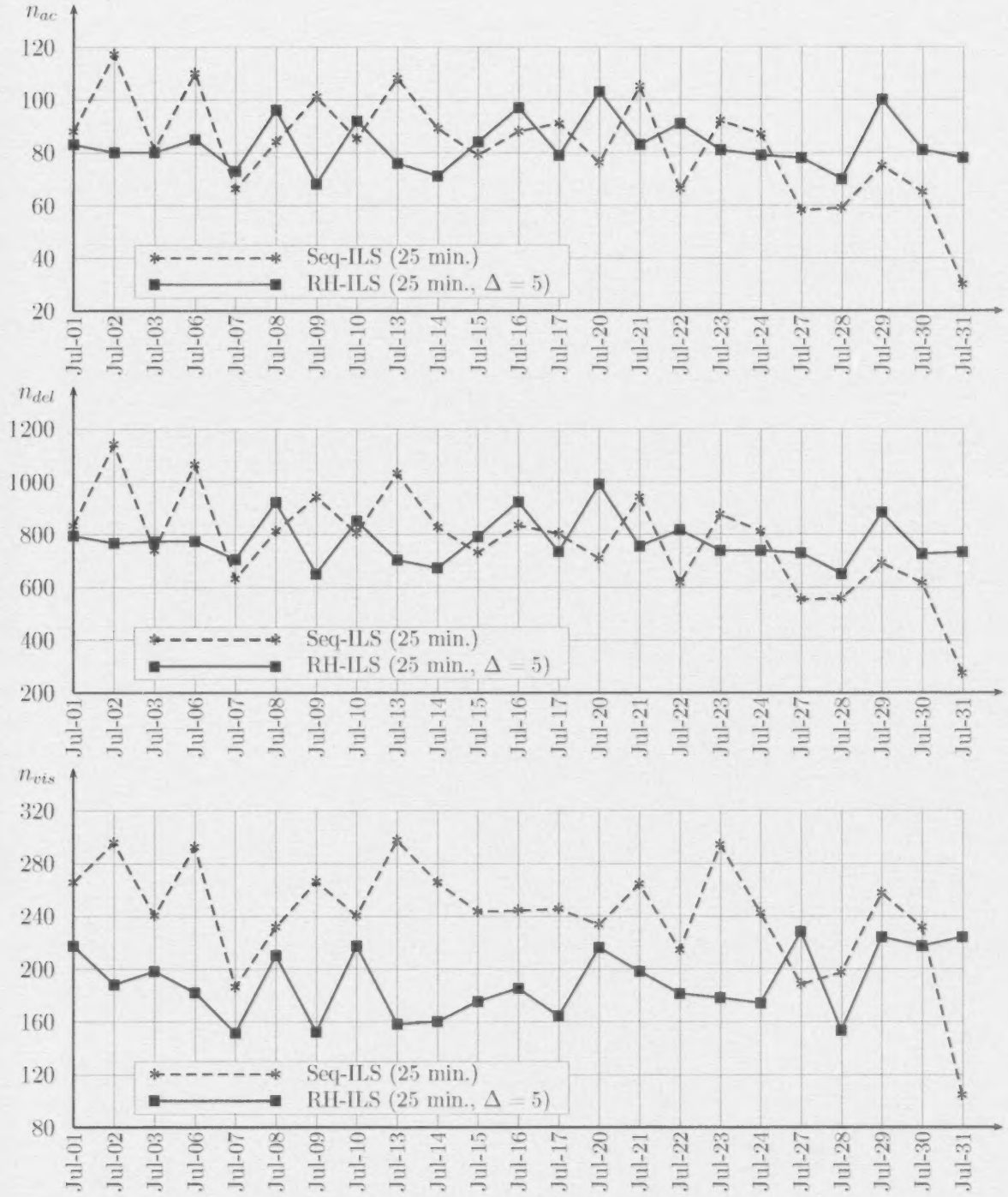

Figure 3: Difference in terms of used auto-carriers (n_{ac}), delivered vehicles (n_{del}), and visits to dealers (n_{vis}) between the Seq-ILS in [11] and the new RH-ILS.

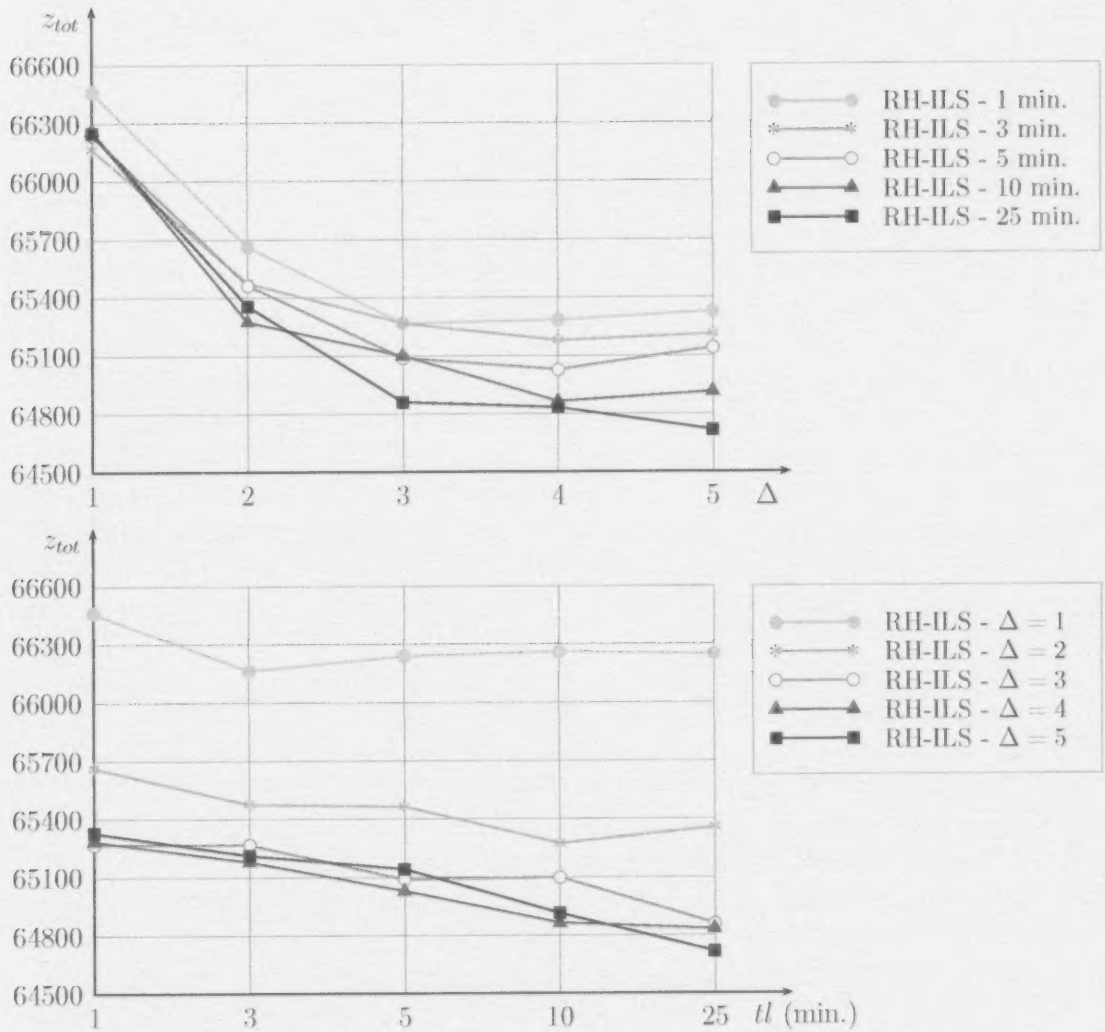

Figure 4: Average z_{tot} values obtained by RH-ILS, by varying Δ (above) or time limit (below).

effort, because the improvements obtained by allowing 25 minutes are very limited.

Table 1 shows some average values, over the 23 working days, for the 25 attempted RH-ILS configurations. Apart from the values of Δ, tl, and z_{tot}, the table provides some insight in the behavior of the algorithm by reporting the impact of the different components of the objective function. The percentage values in the table are obtained by setting $\%z_{km} = 100 \cdot z_{km}/z_{tot}$, $\%z_{fix} = 100 \cdot z_{fix}/z_{tot}$, etc. The table also shows the average number of CPU seconds and ILS iterations required to find the lowest cost solution, sec_z and it_z, respectively, and the average number of ILS iterations performed when reaching the time limit, it_{tot}.

The percentage impacts of the four components of the objective function are quite stable. The traveling costs, z_{km}, take about two thirds of z_{tot}, whereas the auto-carrier fixed costs, z_{fix}, amount to roughly 25% of z_{tot}. The service costs, z_{ser}, usually amount to less than 8%, and the penalty costs for delays, z_{del}, are negligible. It is worth noting that z_{del} are on average 1.35% when Δ=1, but then decrease to below 1% for larger Δ values, showing again that Δ=1 is not advisable. The values taken by sec_z are roughly 75% of the allowed time limits. Similarly, it_z is about 75% of it_{tot}. The total number of ILS iterations increases quite rapidly when $\Delta = 1$, reaching almost 2700

Δ	tl(min.)	z_{tot}	$\%z_{km}$	$\%z_{fix}$	$\%z_{ser}$	$\%z_{del}$	sec_z	it_z	it_{tot}
1	1	66459	65.9%	25.2%	7.6%	1.3%	41	74	107
	3	66165	66.0%	25.2%	7.6%	1.3%	131	251	334
	5	66239	66.0%	25.2%	7.6%	1.3%	218	396	530
	10	66261	65.9%	25.1%	7.6%	1.4%	439	804	1092
	25	66248	65.8%	25.1%	7.6%	1.4%	1050	1914	2682
2	1	65663	66.5%	25.4%	7.9%	0.2%	44	20	29
	3	65476	66.4%	25.4%	7.9%	0.2%	113	58	89
	5	65464	66.4%	25.4%	7.9%	0.2%	218	104	141
	10	65274	66.4%	25.5%	7.9%	0.2%	471	240	305
	25	65356	66.4%	25.4%	7.9%	0.3%	986	512	761
3	1	65268	66.3%	25.5%	8.0%	0.2%	51	13	17
	3	65267	66.4%	25.5%	8.0%	0.1%	119	41	54
	5	65088	66.4%	25.5%	7.9%	0.1%	221	62	79
	10	65098	66.3%	25.6%	8.0%	0.1%	451	126	158
	25	64860	66.3%	25.6%	8.0%	0.1%	1068	287	392
4	1	65283	66.4%	25.5%	8.0%	0.1%	50	15	18
	3	65178	66.3%	25.5%	8.0%	0.1%	138	39	47
	5	65027	66.3%	25.6%	8.0%	0.2%	229	50	66
	10	64863	66.3%	25.6%	8.0%	0.1%	406	95	151
	25	64831	66.3%	25.6%	8.1%	0.1%	1152	298	367
5	1	65327	66.4%	25.5%	8.0%	0.1%	41	11	16
	3	65211	66.3%	25.6%	8.0%	0.1%	111	27	47
	5	65141	66.4%	25.6%	8.0%	0.1%	242	65	76
	10	64912	66.3%	25.6%	8.0%	0.2%	359	109	154
	25	64716	66.2%	25.6%	7.8%	0.4%	967	256	333

Table 1: Evaluation of the average objective function components and algorithmic performances for the 25 attempted RH-ILS configurations.

iterations when tl=25 minutes. For the larger values of Δ, it_{tot} increases less sharply, and never reaches 1000 iterations, proving again that the value of Δ has a strong influence on the ILS.

Table 2 shows detailed solution values for the RH-ILS having $\Delta = 4$ and running for tl=10 minutes, which we consider to be, on the basis of the above comments, a good configuration for our algorithm. The first three columns report the name of the instance, and the number $\overline{n}$ (respectively $\overline{m}$) of dealer requests (respectively vehicle requests) that have been revealed at the beginning of that day. Then, n_{ac} gives the number of used auto-carriers, n_{del} the number of delivered vehicles, and n_{vis} the number of visits to dealers. The remaining columns are the same as those previously discussed in Table 1, but refer to single working days. For each column we also report, at the bottom of the table, minimum, average, and maximum values.

Despite the fact that the number of vehicle requests has some important peaks (for example for Jul-01, Jul-02, and Jul-10), the behavior of the RH-ILS is quite stable. On a daily basis, it uses on average 83 auto-carriers to deliver 774 vehicles to 172 dealers. It provides solutions where all deliveries are performed on time for the first 11 days, before some limited penalties are incurred. The fleet of auto-carriers travels for about 65,000 km per day, with a peak on Jul-22, where it

name	$\overline{n}$	$\overline{m}$	n_{ac}	n_{vis}	n_{del}	z_{tot}	$\%z_{km}$	$\%z_{fix}$	$\%z_{ser}$	$\%z_{del}$	sec_z	it_z	it_{tot}
Jul-01	387	1778	85	198	803	58744	61.0%	28.9%	10.1%	0.0%	356.5	81	144
Jul-02	190	1081	83	193	794	63048	64.5%	26.3%	9.2%	0.0%	102.9	20	132
Jul-03	183	824	79	166	739	74531	72.1%	21.2%	6.7%	0.0%	260.7	58	136
Jul-06	183	856	78	165	740	66717	69.3%	23.3%	7.4%	0.0%	374.3	73	115
Jul-07	170	779	84	157	793	61250	64.9%	27.4%	7.7%	0.0%	114.9	25	114
Jul-08	187	739	75	144	714	54245	64.4%	27.7%	8.0%	0.0%	305.5	51	99
Jul-09	182	882	93	201	892	69486	64.5%	26.8%	8.7%	0.0%	260.0	44	105
Jul-10	204	1031	78	158	725	64805	68.6%	24.1%	7.3%	0.0%	535.2	97	109
Jul-13	181	860	88	182	815	67763	66.0%	26.0%	8.1%	0.0%	575.0	85	89
Jul-14	202	872	75	126	682	51714	63.7%	29.0%	7.3%	0.0%	543.8	73	81
Jul-15	179	903	100	185	951	75414	66.1%	26.5%	7.4%	0.0%	553.6	58	63
Jul-16	198	663	74	137	720	60707	68.7%	24.4%	6.8%	0.1%	541.0	50	55
Jul-17	197	890	78	143	714	52102	61.7%	29.9%	8.2%	0.1%	514.4	49	57
Jul-20	173	744	95	173	890	73256	66.7%	25.9%	7.1%	0.3%	581.4	60	62
Jul-21	187	764	89	200	831	77416	68.9%	22.9%	7.8%	0.5%	359.6	43	74
Jul-22	189	643	99	188	895	80630	68.2%	24.6%	7.0%	0.2%	560.6	66	71
Jul-23	187	728	91	177	843	67854	65.0%	26.8%	7.8%	0.4%	544.7	74	82
Jul-24	173	748	83	170	742	65007	66.4%	25.5%	7.8%	0.2%	237.5	40	102
Jul-27	158	511	80	199	755	67593	67.4%	23.7%	8.8%	0.1%	575.4	136	143
Jul-28	159	532	70	150	652	54937	66.3%	25.5%	8.2%	0.1%	154.6	45	179
Jul-29	173	555	65	126	576	46977	64.1%	27.7%	8.0%	0.2%	379.8	125	194
Jul-30	101	275	85	196	752	57673	60.1%	29.4%	10.2%	0.3%	559.5	241	257
Jul-31	42	146	85	229	786	79983	70.1%	21.3%	8.6%	0.0%	341.0	583	1020
min	42	146	65	126	576	46977	60.1%	21.2%	6.7%	0.0%	102.9	20	55
avg	182	774	83	172	774	64863	66.0%	25.9%	8.0%	0.1%	405.7	95	151
max	387	1778	100	229	951	80630	72.1%	29.9%	10.2%	0.5%	581.4	583	1020

Table 2: Detailed results for RH-ILS with $\Delta = 4$ and 10 CPU minutes of time limit.

travels for slightly more than 80,000 km. The number of elapsed iterations is quite small with the exception of Jul-31, where the number of revealed requests is very small. The algorithm requires on average 400 seconds to reach the best solution, but in some cases it uses almost the entire time limit (as for Jul-13 and Jul-20) showing that there is still some room for finding improved values.

5 Conclusions

We have introduced a new algorithm to plan the delivery of vehicles to dealers by auto-carriers over a multiple-day planning horizon. This algorithm can handle a dynamic demand, a heterogeneous fleet of vehicles, and operational constraints related to the loading of the vehicles on the auto-carriers. It also considers a realistic objective function comprising terms related to traveled distances, vehicle fixed costs, service costs, and penalties for late deliveries. The algorithm is based on a previously proposed iterated local search heuristic which is embedded into a rolling horizon framework. Two new operators have been introduced to deal with the multiple-day planning horizon. The algorithm also incorporates a mechanism to balance demand and vehicle usage in a dynamic setting.

Computational results on data from a major logistics service provider show that considering a planning horizon of 3 to 5 days within a rolling horizon is far superior to the sequential solution of independent daily problems as it leads to a significant decrease in traveled distance. This behavior

is explained by a more balanced workload and by a reduced number of visits to the dealers which are made possible by the less myopic approach. This improved performance of the rolling horizon algorithm is also accompanied by a decrease in total computing time compared with the sequential approach. Experiments with the full objective function show that a rolling horizon of 3 to 5 days usually leads to a significant improvement with respect to shorter horizons of just one or two days. Finally, the algorithm is not too sensitive to the computing time limit and usually produces very good results within just 10 minutes of computing time per day. An interesting extension to this work would consist in taking advantage of stochastic information on future demands to further improve transportation planning.

Acknowledgements

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 227837-09 and by CAPES/Brazil under Grant PVE no. A007/2013. This support is gratefully acknowledged.

References

[1] G.Y. Agbegha. *An Optimization Approach to the Auto-Carrier Problem.* PhD thesis, Case Western Reserve University, 1992.

[2] G.Y. Agbegha, R.H. Ballou, and K. Mathur. Optimizing auto-carrier loading. *Transportation Science*, 32:174–188, 1998.

[3] M. Albareda-Sambola, E. Fernández, and G. Laporte. The dynamic multiperiod vehicle routing problem with probabilistic information. *Computers & Operations Research*, 2014. Forthcoming.

[4] ANFIA. The world automotive industry in 2012. Available online at `http://www.anfia.it/allegati_contenuti/2012_INDUSTRIA_AUTOMOTIVE_MONDIALE_def.pdf`, Italian Association of the Automotive Industry, 2012.

[5] E. Angelelli, N. Bianchessi, R. Mansini, and M.G. Speranza. Short term strategies for a dynamic multi-period routing problem. *Transportation Research Part C*, 17:106–119, 2009.

[6] E. Angelelli, M.G. Speranza, and M.W.P. Savelsbergh. Competitive analysis for dynamic multiperiod uncapacitated routing problems. *Networks*, 49:308–317, 2007.

[7] J.F. Bard, L. Huang, P. Jaillet, and M. Dror. A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science*, 32:189–203, 1998.

[8] N. Bostel, P. Dejax, P. Guez, and F. Tricoire. Multiperiod planning and routing on a rolling horizon for field force optimization logistics. In B.L. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem*, pages 503–525. Springer, New York, 2008.

[9] J.-F. Cordeau, M. Iori, G. Laporte, and J.J. Salazar González. Branch-and-cut for the pickup and delivery traveling salesman problem with LIFO loading. *Networks*, 55:46–59, 2010.

[10] M. Cuadrado and V. Griffin. Mathematical models for the optimization of new vehicle distribution in venezuela (case: Clover international c.a.). *Ingeniería Industrial. Actualidad y Nuevas Tendencias*, 1:53–65, 2009.

[11] M. Dell'Amico, S. Falavigna, and M. Iori. Optimization of a real-world auto-carrier transportation problem. *Transportation Science*, 2014. Forthcoming.

[12] P. Francis and K. Smilowitz. Modeling techniques for periodic vehicle routing problems. *Transportation Research Part B*, 40:872–884, 2006.

[13] P. Hansen, N. Mladenović, and J.A. Moreno Pérez. Variable neighbourhood search: methods and applications. *4OR*, 6:319–360, 2008.

[14] M. Iori and S. Martello. Routing problems with loading constraints. *TOP*, 18:4–27, 2010.

[15] M. Iori and S. Martello. An annotated bibliography of combined routing and loading problems. *Yugoslav Journal of Operations Research*, 23:311–326, 2013.

[16] P. Jaillet, J.F. Bard, L. Huang, and M. Dror. Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science*, 36:292–300, 2002.

[17] M. Jin, S.D. Eksioglu, B. Eksioglu, and H. Wang. Mode selection for automotive distribution with quantity discounts. *Networks and Spatial Economics*, 10:1–13, 2010.

[18] C.-H. Lin. An exact solving approach to the auto-carrier loading problem. *Journal of Society for Transportation and Traffic Studies*, 1:93–106, 2010.

[19] H.R. Lourenço, O.C. Martin, and T. Stützle. Iterated local search: Framework and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics, second edition*, volume 146 of *International Series in Operations Research & Management Science*, pages 363–398. Springer, Berlin, 2010.

[20] B.M. Miller. *Auto Carrier Transporter Loading and Unloading Improvement*. PhD thesis, Air Force Institute of Technology, 2003.

[21] S. Mitrović-Minić, R. Krishnamurti, and G. Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38:669–685, 2004.

[22] V. Pillac, M. Gendreau, C. Guéret, and A.L. Medaglia. A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225:1–11, 2013.

[23] H.N. Psaraftis. Dynamic vehicle routing problems. In B. Golden and A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 223–248. Elsevier, 1988.

[24] R. Tadei, G. Perboli, and F. Della Croce. A heuristic algorithm for the auto-carrier transportation problem. *Transportation Science*, 36:55–62, 2002.

[25] M. Wen, J.-F. Cordeau, G. Laporte, and J. Larsen. The dynamic multi-period vehicle routing problem. *Computers & Operations Research*, 37:1615–1623, 2010.